ELSEVIER

Theoretical Computer Science 255 (2001) 423–436

Theoretical Computer Science

www.elsevier.com/locate/tcs

Generalized scans and tridiagonal systems ☆

Paul F. Fischer [a,1], Franco P. Preparata [b,2,3], John E. Savage [b,*,3,4]

[a] *Mathematics and Computer Science Division, Argonne National Laboratory, USA*
[b] *Department of Computer Science, Brown University, P.O. Box 1910, Providence RI 62912-1910, USA*

Received July 1998; revised July 1999
Communicated by M. Nivat

Abstract

Motivated by the analysis of known parallel techniques for the solution of linear tridiagonal system, we introduce **generalized scans**, a class of recursively defined length-preserving, sequence-to-sequence transformations that generalize the well-known prefix computations (scans). Generalized scan functions are described in terms of three algorithmic phases, the reduction phase that saves data for the third or expansion phase and prepares data for the second phase which is a recursive invocation of the same function on one fewer variable. Both the reduction and expansion phases operate on bounded number of variables, a key feature for their parallelization. Generalized scans enjoy a property, called here protoassociativity, that gives rise to ordinary associativity when generalized scans are specialized to ordinary scans. We show that the solution of positive-definite block tridiagonal linear systems can be cast as a generalized scan, thereby shedding light on the underlying structure enabling known parallelization schemes for this problem. We also describe a variety of parallel algorithms including some that are well known for tridiagonal systems and some that are much better suited to distributed computation.

Keywords: Parallel computation; Prefix; Scan; Numerical computation; Tridiagonal linear system

☆ A preliminary version of this paper appeared in STACS'95, Procs. 12th Ann. Symp. on Theoretical Aspects of Computer Science, Munich, March 2–4 (1995) pp.169–180.

* Corresponding author.
E-mail address: jes@cs.brown.edu (J.E. Savage).

[1] Supported in part by NSF Grant ACS-9405403 and AFOSR Grant F49620-95-1-0074.
[2] Supported in part by NSF Grant CCR-9400232.
[3] Supported in part by the Office of Naval Research under contract N00014-91-J-4052, ARPA Order 8225.
[4] Supported in part by NSF Grant MIP-902570.

PII: S0304-3975(99)00311-4

1. Introduction

The original motivation for this paper were some intriguing questions arising in the parallel solution of tridiagonal systems of linear equations, a problem fundamental in its own right and for its bearing on the solution of banded systems (naturally viewed as block tridiagonal systems). Over the years, a number of direct factorization methods amenable to efficient parallelization have been developed. They include Stone's "scan-based" (or "recursive doubling") algorithm [10], "odd–even cyclic reduction" [3, 11–13], and "partitioning" [5, 14]. These methods reveal various enabling factors of NC-parallelization, which we now briefly review.

Stone's algorithm [10] solves a tridiagonal system with coefficient matrix A by inverting the diagonal matrix D arising from the LDU decomposition of A. This decomposition is obtained through the solution of a linear recurrence of second order in the following two cases: either (i) the base ring is commutative (a property absent in the block-tridiagonal case), or (ii) all upper or lower off-diagonal terms are invertible (a strong condition, especially in the block case). Since analysis of such recurrences reveals an underlying semigroup and its solution is given by the computation of the prefixes over such semigroup, the well-known NC-parallelizability of the latter (scan computation [6]) yields a fast algorithm.

Cyclic (or even–odd) reduction of a tridiagonal system successively eliminates and renumbers the even- or odd-numbered variables. After a single elimination step, the resultant system is again tridiagonal in the remaining variables, which can be renumbered to yield a new system of half the original size. The elimination and renumbering of variables can proceed in parallel because the odd- (or even-) numbered equations are only indirectly coupled. Block cyclic-reduction (BCR) has also been studied as a means of extending this approach to banded systems [4]. Here there is no apparent semigroup operation enabling the emerging parallelism that imposes any additional constraint on the matrices beyond positive-definiteness.

A related parallel technique is represented by partitioning, or substructuring, algorithms [14, 2, 7, 5]. When a system of n equations of bandwidth $2s+1$ is to be solved on a p-processor parallel computer, the diagonal band of the coefficient matrix is partitioned as a block-tridiagonal system of order $2p-1$. Specifically, the block matrices on the diagonal are of two sizes: p large blocks, or "substructures", of size $m=\lceil((n-s(p-1))/p\rceil$ and $p-1$ blocks of size s called "separators". A first phase eliminates the substructures: Because they are coupled only via the separators, the system associated with each substructure can be factored simultaneously to eliminate blocks directly above and below the substructures. Updates to the separator elements during this initial phase are additive and therefore may be asynchronous. The remaining small block-tridiagonal system of order $(p-1)$, with blocks of order s, can be solved, for example, by **BCR**. Thus substructuring is equivalent to BCR if one takes the substructure size equal to the separator size, which for the tridiagonal case, implies $m=s=1$.

The intriguing question was the identification of the underlying connections among these parallelizing techniques. We have succeeded in reformulating the general prob-

lem as a length-preserving transformation of input sequences to output sequences of which the ordinary prefix computations (also known as *scans*) are very special and important cases. For this reason we have named such transformation *generalized scans*. A very interesting feature of the generalized-scan operators is a property, called here **protoassociativity**, which becomes ordinary associativity on the set of suitably defined (fixed-length) strings of input symbols (a semigroup).

It should be emphasized that in this paper we adopt a purely algebraic viewpoint because our intent is to elucidate the algebraic structure that enables parallelism. Therefore we are not concerned with the numerical behavior of the techniques, such as stability, and we do not consider iterative methods for which noteworthy solutions [9, 8] for tridiagonal systems have been presented in the literature.

The paper is organized as follows. In Section 2 we introduce the notion of generalized scans and verify that ordinary scans are a special case of them. Next, in Section 3 we discuss the characteristic property of generalized scans (protoassociativity) and relate it to ordinary associativity. Finally in Section 4 we cast the solution of a block tridiagonal system as a generalized scan thereby revealing its inherent parallelism, which had been partially exhibited by the algorithms appearing in the literature.

2. Generalized scans

After briefly reviewing the standard scan operation, we introduce the generalized scan operation.

Let $\boldsymbol{S}=(S,\oplus)$ denote a semigroup consisting of a set S and an associative operator $\oplus: S^2 \to S$. The **scan function** $f_{\text{scan}}^{(n)}: S^n \to S^n$ maps the sequence $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ to the sequence $\boldsymbol{y}=(y_1, y_2, \ldots, y_n)$, that is, $f_{\text{scan}}^{(n)}(\boldsymbol{x})=\boldsymbol{y}$ where

$$y_i = x_1 \oplus x_2 \oplus \cdots \oplus x_i, \quad i = 1, 2, \ldots, n.$$

The function $f_{\text{scan}}^{(n)}$ can be computed on a parallel machine in time $O(\log n)$ with (optimal) work proportional to n [6]. Scan computations have been used to solve a large variety of computational problems [1].

Below we recursively define the (α,β)-generalized scan function on n inputs, $f_{\text{gen\_scan}}^{(n,\alpha,\beta)}(\boldsymbol{x})$, which produces n outputs. When $n=\alpha+\beta$ (the bottom of the recursion), $f_{\text{gen\_scan}}^{(n,\alpha,\beta)}(\boldsymbol{x})=T(\boldsymbol{x})$, where T is a fixed function. For larger values of n, a generalized scan is "centered" on an arbitrary position of $\boldsymbol{x}$ whose index, j, is preceded by at least α and followed by at least β inputs (call this the set $\boldsymbol{x}_{[j-\alpha,j+\beta]}$). In the **reduction phase** the $\alpha+\beta+1$ terms in $\boldsymbol{x}_{[j-\alpha,j+\beta]}$ are replaced by the $\alpha+\beta$ values of a function R (independent of j) acting on $\boldsymbol{x}_{[j-\alpha,j+\beta]}$. This has the effect of suppressing one input. Also, a new value, c_j, called the **record at the jth position** (the position on which the scan is centered), is computed by a function ρ on $\boldsymbol{x}_{[j-\alpha,j+\beta]}$. Next, the new set of $n-1$ values obtained by replacing $\boldsymbol{x}_{[j-\alpha,j+\beta]}$ by the values of R is recursively processed (in the **recursive phase**) to produce the $(n-1)$-tuple $\boldsymbol{y}$. In the **expansion phase** the first $j-\alpha-1$ components and last $n-j-\beta$ components of $\boldsymbol{y}$ are passed directly to the output without

change in the corresponding output positions. The remaining $\alpha+\beta+1$ components of the n outputs are obtained by applying an expansion function, $f^{(n,\alpha,\beta)}_{\text{expansion},j}(\hat{\boldsymbol{y}},c_j)$, to $\hat{\boldsymbol{y}}$, the remaining $\alpha+\beta$ components of $\boldsymbol{y}$ and c_j, the record at position j.

Definition 1. For sets S_{IN}, S_{OUT}, and S_{M} and nonnegative integers α and β, $\alpha+\beta>0$, let $R: S_{\text{IN}}^{\alpha+\beta+1}\to S_{\text{IN}}^{\alpha+\beta}$, $\rho: S_{\text{IN}}^{\alpha+\beta+1}\to S_{\text{M}}$, $T: S_{\text{IN}}^{\alpha+\beta}\to S_{\text{OUT}}^{\alpha+\beta}$, and $E: S_{\text{OUT}}^{\alpha+\beta}\times S_{\text{M}}\to S_{\text{OUT}}^{\alpha+\beta+1}$ be four functions.

For $\alpha+1\leqslant j\leqslant n-\beta$ let $f^{(n,\alpha,\beta)}_{\text{reduction},j}: S_{\text{IN}}^{n}\to S_{\text{IN}}^{n-1}\times S_{\text{M}}$ be defined as

$$f^{(n,\alpha,\beta)}_{\text{reduction},j}(\boldsymbol{x})=(\boldsymbol{r},c_j)=(r_1,r_2,\ldots,r_{j-1},r_{j+1},\ldots,r_n;c_j),$$

where

$$\begin{aligned}
&r_i=x_i \quad \text{for} \quad i<j-\alpha \text{ and } i>j+\beta,\\
&(r_{j-\alpha},\ldots,r_{j-1},r_{j+1},\ldots,r_{j+\beta})=R(x_{j-\alpha},\ldots,x_{j+\beta}),\\
&c_j=\rho(x_{j-\alpha},\ldots,x_{j+\beta}).
\end{aligned}$$

Let $f^{(n,\alpha,\beta)}_{\text{expansion},j}: S_{\text{OUT}}^{n-1}\times S_{\text{M}}\to S_{\text{OUT}}^{n}$ be defined by

$$f^{(n,\alpha,\beta)}_{\text{expansion},j}(\hat{y}_1,\ldots,\hat{y}_{j-1},\hat{y}_{j+1},\ldots,\hat{y}_n;c_j)=(y_1,y_2,\ldots,y_n),$$

where

$$\begin{aligned}
&y_i=\begin{cases}\hat{y}_i & \text{for } i<j-\alpha,\\ \hat{y}_{i-1} & \text{for } i>j+\beta\end{cases}\\
&(y_{j-\alpha},\ldots,y_{j+\beta})=E(\hat{y}_{j-\alpha},\ldots,\hat{y}_{j-1},\hat{y}_{j+1},\ldots,\hat{y}_{j+\beta};c_j).
\end{aligned}$$

An (α,β)-**generalized scan** function $f^{(n,\alpha,\beta)}_{\text{gen\_scan}}: S_{\text{IN}}^{n}\to S_{\text{OUT}}^{n}$ is a length-preserving transformation of an input sequence $\boldsymbol{x}$ to an output sequence $\boldsymbol{y}=f^{(n,\alpha,\beta)}_{\text{gen\_scan}}(\boldsymbol{x})$ defined below where $f^{(n,\alpha,\beta)}_{\text{reduction},n-\beta}(\boldsymbol{x})=(\boldsymbol{r},c_{n-\beta})$.

$$f^{(n,\alpha,\beta)}_{\text{gen\_scan}}(\boldsymbol{x})=\begin{cases}T(\boldsymbol{x}) & \text{when } n=(\alpha+\beta),\\ f^{(n,\alpha,\beta)}_{\text{expansion},n-\beta}(f^{(n-1,\alpha,\beta)}_{\text{gen\_scan}}(\boldsymbol{r});c_{n-\beta}) & \text{when } n\geqslant\alpha+\beta+1.\end{cases}$$

The scan function $f^{(n)}_{\text{scan}}$ is an instance of a generalized scan function for which $\alpha=\beta=1$, $S_{\text{IN}}=S_{\text{OUT}}=S_{\text{M}}=S$, $\oplus: S\times S\to S$ is an associative operation (for all $a,b,c\in S_{\text{IN}}$, $a\oplus(b\oplus c)=(a\oplus b)\oplus c$), and

$$\begin{aligned}
&R(u_1,u_2,u_3)=(u_1,u_2\oplus u_3),\\
&\rho(u_1,u_2,u_3)=u_2,\\
&E(v_1,v_3;u_2)=(v_1,v_1\oplus u_2,v_3),\\
&T(u_1,u_2)=(u_1,u_1\oplus u_2).
\end{aligned}$$

To see that this interpretation of a standard scan computation is correct, note that it is correct for $n=2$ and for $n\geqslant 3$ after the reduction phase, a scan computation is done on

the sequence $(x_1,\ldots,x_{n-2},x_{n-1}\oplus x_n)$ which provides $y_i = x_1 \oplus \cdots \oplus x_i$ for $1 \leqslant i \leqslant n-2$ and $y_n = x_1 \oplus \cdots \oplus x_n$. The carry from the reduction phase is $c_{n-\beta} = c_{n-1} = x_{n-1}$ which is combined with y_{n-2} to compute y_{n-1}.

Let $R^{(j)}$ and $E^{(j)}$ denote reduction and expansion on the jth index and let $F^{[s]}$ denote the (α,β)-generalized scan function after suppression of the indices in the sequence $\boldsymbol{s}$ in the order given in $\boldsymbol{s}$. Thus, $F^{[\varepsilon]}$ denotes the (α,β)-generalized scan function on all n inputs (i.e., sequence $\boldsymbol{s}$ is the empty sequence ε). An (α,β)-generalized scan function is defined recursively by the following expression where the functions are applied in left-to-right order:

$$F^{[\varepsilon]} = R^{(n-\beta)}F^{[(n-\beta)]}E^{(n-\beta)}.$$

It follows that we can write $F^{[\varepsilon]}$ as follows:

$$F^{[\varepsilon]} = R^{(n-\beta)}R^{(n-\beta-1)}\cdots R^{(\alpha+1)}TE^{(\alpha+1)}\cdots E^{(n-\beta-1)}E^{(n-\beta)},$$

where $T = F^{[\boldsymbol{q}]}$ and $\boldsymbol{q} = (n-\beta, n-\beta-1,\ldots,\alpha+1)$. When $\boldsymbol{s} = (n-\beta,\ldots,n-\beta-t)$ this implies that we can write $F^{[\boldsymbol{s}]}$ as follows:

$$F^{[\boldsymbol{s}]} = R^{(n-\beta-t-1)}F^{[(n-\beta,\ldots,n-\beta-t-1)]}E^{(n-\beta-t-1)}.$$

This recursive definition of an (α,β)-generalized scan function requires that reductions be performed on inputs in decreasing order of their index. We shall see later that, when a specific condition is satisfied, the indices can be suppressed in any order.

The quadruple (R,ρ,E,T) is called the **generalized scan operator**. The set of $\alpha+\beta+1$ arguments of R and ρ is called the *support* of the operator. To define minimal values of α and β, we stipulate that R effectively depends upon each variable of its support. (*R depends on variable x_j* if there are values for the other variables such that a change in x_j causes a change in the output.)

Since a generalized scan function passes inputs outside the support of R and E to its outputs without alteration, as shown below, if the following condition applies, the order of two consecutive reductions and subsequent expansions can be exchanged.

Definition 2. Let $R = (R_1,R_2,\ldots,R_{\alpha+\beta})$ and $E = (E_1,E_2,\ldots,E_{\alpha+\beta+1})$, respectively, where $R_j : S_{\mathrm{IN}}^{\alpha+\beta+1} \to S_{\mathrm{IN}}$ and $E_j : S_{\mathrm{OUT}}^{\alpha+\beta} \times S_{\mathrm{M}} \to S_{\mathrm{OUT}}$. The generalized scan operator (R,ρ,E,T) is *protoassociative* if for all $x_1,\ldots,x_{\alpha+\beta+2} \in S_{\mathrm{IN}}$ and $1 \leqslant j \leqslant \alpha+\beta$

$$\begin{aligned} &R_j(x_1,R_1(x_2,\ldots,x_{\alpha+\beta+2}),\ldots,R_{\alpha+\beta}(x_2,\ldots,x_{\alpha+\beta+2})) \\ &\quad = R_j(R_1(x_1,\ldots,x_{\alpha+\beta+1}),\ldots,R_{\alpha+\beta}(x_1,\ldots,x_{\alpha+\beta+1}),x_{\alpha+\beta+2}) \end{aligned} \tag{1}$$

and for all $y_1,\ldots,y_{\alpha+\beta+1} \in S_{\mathrm{OUT}}$ and $2 \leqslant j \leqslant \alpha+\beta$,

$$\begin{aligned} &E_j(E_2(y_1,\ldots,y_{\alpha+\beta};r_4),\ldots,E_{\alpha+\beta+1}(y_1,\ldots,y_{\alpha+\beta};r_4);r_2), \\ &\quad = E_j(E_1(y_1,\ldots,y_{\alpha+\beta};r_3),\ldots,E_{\alpha+\beta}(y_1,\ldots,y_{\alpha+\beta};r_3);r_1), \end{aligned} \tag{2}$$

where $r_1 = \rho(x_1,\ldots,x_{\alpha+\beta+1})$, $r_2 = \rho(x_2,\ldots,x_{\alpha+\beta+2})$, $r_3 = \rho(R(x_1,\ldots,x_{\alpha+\beta+1}),x_{\alpha+\beta+2})$, and $r_4 = \rho(x_1,R(x_2,\ldots,x_{\alpha+\beta+2}))$.

As shown below, this condition implies that the value of a generalized scan function is the same for any reduction order and corresponding expansion order.

Theorem 3. *Let* $n \geqslant (\alpha+\beta+2)$. *Given an* (α,β)*-generalized scan function* $f^{(n,\alpha,\beta)}_{\text{gen\_scan}} : S^n_{\text{IN}} \to S^n_{\text{OUT}}$ *whose generalized scan operator* (R,ρ,E,T) *is protoassociative, all orders of suppression of indices yield the same value for* $f^{(n,\alpha,\beta)}_{\text{gen\_scan}}$. *That is,*

$$F^{[\varepsilon]} = R^{(i_1)} \cdot R^{(i_2)} \cdots R^{(i_{n-\alpha-\beta})} T E^{(i_{n-\alpha-\beta})} \cdots E^{(i_2)} \cdot E^{(i_1)}$$

for every permutation $(i_1, i_2, \ldots, i_{n-\alpha-\beta})$ *of* $(\alpha+1, \alpha+2, \ldots, n-\beta)$.

Proof. From Eq. (1) the reduction by suppression of index $(n-\beta)$ followed by index $(n-\beta-1)$ produces the same input to $f^{(n-2,\alpha,\beta)}_{\text{gen\_scan}}$ as does the suppression of index $(n-\beta-1)$ followed by index $(n-\beta)$. Also, from (2) expansion on index $(n-\beta)$ followed by index $(n-\beta-1)$ produces the same output as does expansion on index $(n-\beta-1)$ followed by index $(n-\beta)$. It follows that both reduction/expansion orders produce the same value for the generalized scan function. That is,

$$\begin{aligned} F^{[\varepsilon]} &= R^{(n-\beta)} \cdot R^{(n-\beta-1)} F^{[(n-\beta,n-\beta-1)]} E^{(n-\beta-1)} \cdot E^{(n-\beta)} \\ &= R^{(n-\beta-1)} \cdot R^{(n-\beta)} F^{[(n-\beta,n-\beta-1)]} E^{(n-\beta)} \cdot E^{(n-\beta-1)}. \end{aligned}$$

Because the first reduction/expansion phase can be done on index $(n-\beta-1)$ preceded/followed by a generalized scan function on the remaining indices, it follows by induction that first reduction/expansion can be done on any index and that any reduction/expansion order is permissible, which is the desired conclusion. □

Shown in Fig. 1 is a serial decomposition of the scan function $f^{(6)}_{\text{scan}} = f^{(6,1,1)}_{\text{gen\_scan}}$ using its representation as a generalized scan function. Here reduction is carried out in the following sequence: third, fifth, second, and fourth. At this point the string has length 2 and the bottom of recursion is executed (hexagonal-shaped node). Finally, expansion occurs in the reverse order of reduction. Data move along the edges from left to right. The hexagonal-shaped vertex computes the scan function on two inputs. Dashed lines in Fig. 1 symbolize storage of an element for later use.

The symmetry of reductions and expansions of a generalized scan allows for its realization as a graph obtained by folding the graph about its midpoint and fusing corresponding reduction and expansion vertices, as suggested in Fig. 2. In the folded graph, data move from left to right in the reduction phase to the bottom-of-recursion vertex on the right and then back to the input in the expansion phase.

The preceding correct interpretation of the standard scan does not agree with the usual implementation by means of a tree network. We now illustrate an alternative interpretation that corresponds indeed to a tree computation. We assume that S is a monoid, i.e. it contains the identity element ε (adjunction of the identity is always feasible) with the property $\varepsilon \oplus x = x \oplus \varepsilon = x$. We extend the input sequence $\boldsymbol{x} = (x_1,\ x_2, \ldots, x_n)$ with an identity term to its right to obtain $\boldsymbol{x} = (x_1,\ x_2, \ldots,\ x_n,\ \varepsilon)$ and define $\boldsymbol{y} = (\varepsilon, y_1, y_2, \ldots, y_n)$

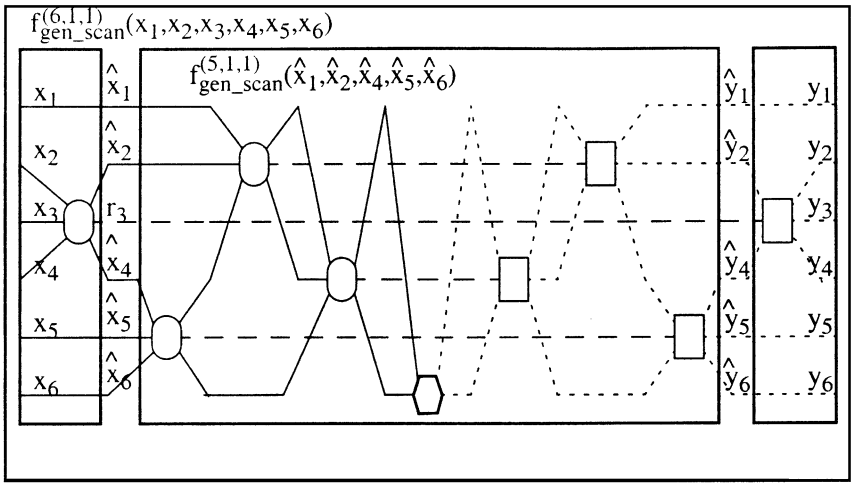

Fig. 1. A realization of a generalized scan on six inputs with $\alpha=\beta=1$ in which the reduction and expansion occur on the third, fifth, second, and fourth inputs, in that order.

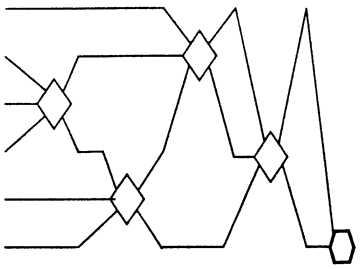

Fig. 2. The graph resulting from folding the network of Fig. 1 about the hexagonal module. Each edge is to be interpreted as a pair of directed arcs with opposite orientation.

to be the *shifted scan* of $\boldsymbol{x}$, with the usual meaning for y_j, $j=1,\ldots,n$. For $\alpha=0$ and $\beta=1$ we define the following functions:

$$R(u_1,u_2)=(u_1\oplus u_2),$$
$$\rho(u_1,u_2)=u_1,$$
$$E(v_1;u_1)=(v_1,v_1\oplus u_1),$$
$$T^{(1)}(u)=\varepsilon.$$

We then have the following recursion for $j\geqslant 1$:

$$f^{(n-1,0,1)}_{\text{reduction},j}(\boldsymbol{x})=(x_1,\ldots,x_{j-1},x_j\oplus x_{j+1},x_{j+2},\ldots,x_n,\varepsilon;x_j)=(\boldsymbol{x}^*,x_j),$$
$$f^{(n-1,0,1)}_{\text{gen\_scan}}(\boldsymbol{x}^*)=(\varepsilon,\hat{y}_1,\ldots,\hat{y}_{j-1},\hat{y}_{j+1},\ldots,\hat{y}_n)$$

Finally the expansion phase computes $y_j = \hat{y}_{j-1} \oplus x_j$ and inserts it among the other outputs thereby yielding $(\varepsilon, y_1, \ldots, y_n)$, as claimed. It is a simple exercise to verify that the graph description of the shifted scan just defined yields the familiar binary tree network.

Next, we examine some important properties of generalized scans, which also characterize their inherent parallelism.

3. Properties of protoassociative generalized scans

In the preceding section we have shown that a schedule of index suppressions represented by a permutation of $(\alpha+1, \alpha+2, \ldots, n-\beta)$ correctly evaluates the generalized scan function.

While the above definition of a generalized scan presents the corresponding computation as serial, a little thought reveals that the definition hides parallelism that can be exploited. A set of successive index suppressions whose domains do not overlap can be executed in parallel. More specifically, we now give an algebraic circuit interpretation of the inherent parallelism of the computation.

We recall that an algebraic circuit is a directed acyclic graph whose vertices represent either variables or functions. The function computed by a circuit is that obtained through the application of functional composition on the functions associated with vertices. The size of an algebraic circuit is the number of vertices associated with functions and its depth is the number of vertices on the longest path from an output vertex to an input vertex.

Theorem 4. *Given the functions* $\{R, \rho, E, T\}$, *the generalized scan function* $f_{\text{gen\_scan}}^{(n,\alpha,\beta)}$ *can be realized by an algebraic circuit of size* $O(n)$ *and depth* $O(\log_\gamma n)$ *where* $\gamma = (\alpha+\beta+1)/(\alpha+\beta)$.

Proof. We give a recursive construction that exhibits parallelism. Let $n_0 = n$. Initially, the indices $j = k(\alpha+\beta+1)-\beta$ for $1 \leqslant k \leqslant m_1$ where $m_1 = \lfloor n_0/(\alpha+\beta+1) \rfloor$ are suppressed (reduction phase). These suppressions can be performed in parallel because the domains and ranges of the reductions do not overlap. This operation is implemented by a single stage of appropriate modules acting in constant time. This step produces a sequence of $n_1 = n_0 - m_1 \leqslant (n+1)/\gamma$ outputs, which become the inputs to a (recursively defined) generalized scan circuit.

On the output of the latter circuit, a single stage of appropriate modules performs in constant time the expansion operation on the same set of indices as the initial reduction. Since the initial step reduces the input size by a factor of γ, it is clear that the depth of the circuit is proportional to $\log_\gamma n$. Moreover, the size, $S(n)$, of the circuit on n inputs satisfies $S(n) \leqslant m_1 + S(n_1) \leqslant n/(\alpha+\beta+1) + S((n+1)/\gamma)$ which is linear in n. □

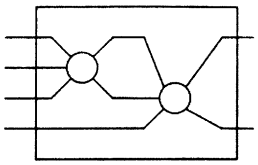

Fig. 3. Reduction steps for a generalized scan on four inputs when $\alpha = \beta = 1$.

Let A_{uniform} be the algorithm used in the proof of Theorem 4. While A_{uniform} makes explicit the parallelism inherent in a generalized scan function, it may be quite inefficient on a network of p serial processors operating in parallel (the **distributed computing model**). On such networks the time to communicate between two processors can be much larger than the time to execute a single instruction. As a consequence, a naive implementation of A_{uniform} can require many more interprocessor communication steps than a more careful implementation. In particular, if the p processors in a network are given a linear order and the variables of $f_{\text{gen\_scan}}^{(n,\alpha,\beta)}$ are assigned to processors consecutively in groups of size n/p when $n/p \gg (\alpha + \beta + 1)$, then in each of the $O(\log_\gamma n)$ parallel phases each pair of adjacent processors must exchange data.

An alternative approach, referred to as $A_{\text{distributed}}$, can reduce the number of phases on which messages are exchanged between processors to $O(\log_\gamma p)$. Since p is typically much smaller than n and the time for each message is large, this can result in a large savings in execution time.

In $A_{\text{distributed}}$, each of the p processors works on its local set of n/p variables as long as possible. All processors apply reduction steps to their n/p variables until at most $\alpha + \beta + 1$ variables remain. Shown in Fig. 3 is the graph associated with the reduction phase of a $(1,1)$-generalized scan on four inputs. It demonstrates that no communication is needed until as many reductions as possible have been performed on one processor. Let $m \leqslant (\alpha + \beta + 1)p$ be the total number of variables remaining on all p processors. At this point the p processors must cooperate through the exchange of messages to compute $f_{\text{gen\_scan}}^{(m,\alpha,\beta)}$. Applying A_{uniform} to this problem requires $O(\log_\gamma m) = O(\log_\gamma p)$ inter-processor communications after which the expansion steps can be performed within each processor without the further exchange of messages. We summarize these observations below.

Theorem 5. *Given the functions* $\{R, \rho, E, T\}$, *the generalized scan function* $f_{\text{gen\_scan}}^{(n,\alpha,\beta)}$ *can be realized on a network of* p *linearly connected processors in* $O(n/p)$ *independent parallel steps plus* $O(\log_\gamma p)$ *communication steps. The communication cost ranges between* $O(p)$ *and* $O(\log p)$ *as the communication time becomes independent of the interprocessor distance on the linear array.*

In addition to the illustrated parallelizability, we now discuss an important consequence of the protoassociativity of generalized scans. Consider now an input string of length $3(\alpha + \beta)$, represented through its indices $1, \ldots, 3(\alpha + \beta)$, and suppose we

apply to it an arbitrary schedule of suppressions of indices $\alpha+1, \alpha+2, \ldots, 3\alpha+2\beta$, yielding as output a string of $(\alpha+\beta)$ terms (any such schedule, as a consequence of protoassociativity, yields the same result). Among all equivalent suppression schedules consider the two following ones:

(1) Suppression of indices $\alpha+1, \ldots, 2\alpha+\beta$, *followed* by the suppression of indices $2\alpha+\beta+1, \ldots, 3\alpha+2\beta$;
(2) Suppression of indices $2\alpha+\beta+1, \ldots, 3\alpha+2\beta$, *followed* by the suppression of indices $\alpha+1, \ldots, 2\alpha+\beta$.

Since the two schedules are equivalent, if we consider a string of $\alpha+\beta$ consecutive terms of S_{IN} as an element of a set Σ, and consider the previously defined suppression of $\alpha+\beta$ contiguous indices as a binary operation "$*$" on Σ, the established equivalence reveals that $*$ is associative, i.e., $(\Sigma, *)$ is a semigroup. This observation ties protoassociativity to the standard algebraic notion of associativity, and sheds light on the comparable suitability to parallelization of traditional scans and generalized scans.

As a final remark, if we test the interpretation of scan as a generalized scan expressed by Eq. (1) for protoassociativity, with reference to the function R for a normal scan we find

$$R_1(x_1, x_2, x_3 \oplus x_4) = R_1(x_1, x_2 \oplus x_3, x_4),$$
$$R_2(x_1, x_2, x_3 \oplus x_4) = R_2(x_1, x_2 \oplus x_3, x_4).$$

The first line yields the trivial relation $x_1 = x_1$; the second one, however, yields $x_2 \oplus (x_3 \oplus x_4) = (x_2 \oplus x_3) \oplus x_4$, i.e., standard associativity. An analogous result is obtained referring to function E.

4. Solving block tridiagonal systems

A tridiagonal system of equations over the ring $\boldsymbol{R}$ in the unknowns $y_1, y_2, \ldots, y_n$ is described by the following equations:

$$a_j y_{j-1} + b_j y_j + c_j y_{j+1} = d_j \quad \text{for } 1 \leqslant j \leqslant n,$$

where a_j, b_j, c_j and d_j are in $\boldsymbol{R}$ and coefficients with indices $j \leqslant 0$ or $j \geqslant n+1$ are zero. This set of equations is described by the tridiagonal matrix A of coefficients shown below:

$$A = \begin{bmatrix} b_1 & c_1 & & & & & \\ a_2 & b_2 & c_2 & & & & \\ & a_3 & b_3 & & & & \\ & & & \ddots & c_{j-1} & & \\ & & & a_j & b_j & & \\ & & & & & \ddots & c_{n-1} \\ & & & & & a_n & b_n \end{bmatrix}$$

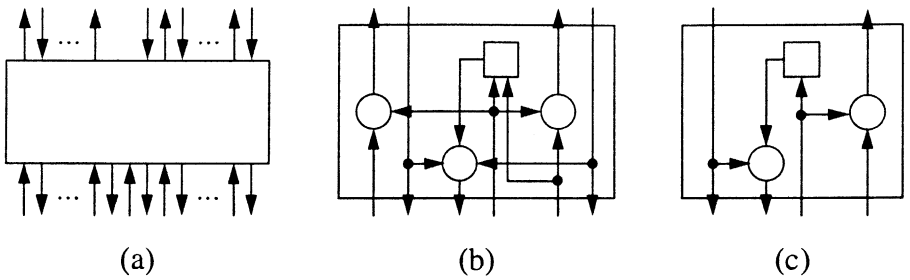

Fig. 4. An associative operator in a generalized scan (illustrated for a tridiagonal system solver).

The system of equations associated with the matrix A is

$$A\boldsymbol{y} = \boldsymbol{x} \tag{3}$$

(where, of course, $\boldsymbol{y}$ and $\boldsymbol{x}$ are column vectors). Without loss of generality we may assume that $\boldsymbol{R}$ is itself a ring of $b \times b$ matrices over a ring of elements.

We are interested in solving such systems when the matrix A is **positive definite** (especially when $\boldsymbol{R}$ is a ring of $b \times b$ matrices). If the matrix A is not positive definite, as is well known, the system $A^{\mathrm{T}}A\boldsymbol{y} = A^{\mathrm{T}}\boldsymbol{x}$ has the same solution as (3) and the matrix $A^{\mathrm{T}}A$ is (symmetric) positive definite (although its bandwidth is twice as large).

The data for this system (3) are conveniently characterized by the following set of n four-tuples (referred to here for convenience as the **lambda notation**):

$$\lambda_i = (a_i, b_i, c_i, d_i), \quad 1 \leqslant i \leqslant n. \tag{4}$$

We now describe a method of factoring a tridiagonal matrix that allows us to describe its solution as the computation of a generalized scan function. To simplify the exposition, we illustrate the procedure for a specific example and follow with the general form. Consider the case $n = 6$:

$$\begin{bmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & a_3 & b_3 & c_3 & & \\ & & a_4 & b_4 & c_4 & \\ & & & a_5 & b_5 & c_5 \\ & & & & a_6 & b_6 \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{pmatrix}.$$

Elimination of the coefficients in the third column (chosen arbitrarily) via one round of Gaussian elimination yields the modified system

$$\begin{bmatrix} b_1 & c_1 & & & & \\ a_2 & \hat{b}_2 & 0 & \hat{c}_2 & & \\ & a_3 & b_3 & c_3 & & \\ & \hat{a}_4 & 0 & \hat{b}_4 & c_4 & \\ & & & a_5 & b_5 & c_5 \\ & & & & a_6 & b_6 \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix} = \begin{pmatrix} d_1 \\ \hat{d}_2 \\ d_3 \\ \hat{d}_4 \\ d_5 \\ d_6 \end{pmatrix} \tag{5}$$

Expressions for the hatted variables (e.g. $\hat{b}_i$) obtained from the Gaussian elimination are given below for the general case. The crucial observation is that (5) is equivalent to the pair of problems

$$\begin{bmatrix} b_1 & c_1 & & & \\ a_2 & \hat{b}_2 & \hat{c}_2 & & \\ & \hat{a}_4 & \hat{b}_4 & c_4 & \\ & & a_5 & b_5 & c_5 \\ & & & a_6 & b_6 \end{bmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix} = \begin{pmatrix} d_1 \\ \hat{d}_2 \\ \hat{d}_4 \\ d_5 \\ d_6 \end{pmatrix} \tag{6}$$

and

$$y_3 = b_3^{-1}(d_3 - a_3 y_2 - c_3 y_4). \tag{7}$$

We see that with one round of Gaussian elimination on an *arbitrary* index, we recover a tridiagonal system (6) with one less equation, to be solved (formally) via recursion, and a single equation for y_3 which is solved subsequently.

For an arbitrary index j the information needed to compute y_j as well as solve the reduced system is captured by the following lambda notation:

$$\hat{\lambda}_i = \begin{cases} (a_{j-1}, \hat{b}_{j-1}, \hat{c}_{j-1}, \hat{d}_{j-1}), & i = j-1, \\ (a_j, b_j^{-1}, c_j, d_j) \text{ (special)}, & i = j, \\ (\hat{a}_{j+1}, \hat{b}_{j+1}, c_{j+1}, \hat{d}_{j+1}), & i = j+1, \\ (a_i, b_i, c_i, d_i) & \text{otherwise} \end{cases} \tag{8}$$

with

$$\begin{aligned} &\hat{b}_{j-1} = b_{j-1} - c_{j-1} b_j^{-1} a_j, \quad \hat{a}_{j+1} = -a_{j+1} b_j^{-1} a_j, \\ &\hat{c}_{j-1} = -c_{j-1} b_j^{-1} c_j, \quad \hat{b}_{j+1} = b_{j+1} - a_{j+1} b_j^{-1} c_j, \\ &\hat{d}_{j-1} = d_{j-1} - c_{j-1} b_j^{-1} d_j, \quad \hat{d}_{j+1} = d_{j+1} - a_{j+1} b_j^{-1} d_j. \end{aligned} \tag{9}$$

After solving the new set of $n-1$ equations to produce the result vector $(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{n-1})$, the original system of equations is solved as follows: these $n-1$ results are passed directly to the output and y_j is computed by combining $\hat{\lambda}_j$ with $\hat{y}_{j-1}$ and $\hat{y}_{j+1}$. If we view y_{j-1}, y_j, and y_{j+1} as computed from $\hat{y}_{j-1}$, $\hat{y}_j$, and $\hat{y}_{j+1}$ in the preceding stage, then

$$\begin{aligned} &y_i = \hat{y}_i \quad \text{for } i \neq j-1, \\ &y_j = b_j^{-1}(d_j - a_j \hat{y}_{j-1} - c_j \hat{y}_j). \end{aligned}$$

We now realize that solving a tridiagonal system of equations amounts to computing a generalized scan function with the following operator (for $\alpha = \beta = 1$):

$$\begin{aligned} &R(\lambda_{j-1}, \lambda_j, \lambda_{j+1}) = (\hat{\lambda}_{j-1}, \hat{\lambda}_{j+1}, ; \rho(\lambda_{j-1}, \lambda_j, \lambda_{j+1})), \\ &\rho(\lambda_{j-1}, \lambda_j, \lambda_{j+1}) = \hat{\lambda}_j, \end{aligned}$$

$$E(\hat{y}_{j-1}, \hat{y}_{j+1}; \hat{\lambda}_j) = (\hat{y}_{j-1}, b_j^{-1}(d_j - a_j\hat{y}_{j-1} - c_j\hat{y}_j), \hat{y}_{j+1}),$$
$$T(\lambda_1, \lambda_n) = ((b_1 - c_1 b_n^{-1} a_n)^{-1}(d_1 - c_1 b_n^{-1} d_n),$$
$$(b_n - a_n b_1^{-1} c_1)^{-1}(d_n - a_n b_1^{-1} d_1)).$$

It is also relatively easy to verify that the above operator is protoassociative.

It is important to note that the parallel solution of block tridiagonal matrices obtained by casting this problem as the solution of a generalized scan computation requires only that the diagonal matrices $b_1, b_2, \ldots, b_n$ be invertible, a property guaranteed by the assumption that the matrix is positive definite, and not that any of the off-diagonal entries be invertible.

We note that the schedule $\boldsymbol{A}_{\text{uniform}}$ corresponds to the well-known cyclic reduction algorithm (e.g., [3, 11–13], while $\boldsymbol{A}_{\text{distributed}}$ corresponds to the partitioning method (e.g., [5, 14]). Both approaches exploit as a common foundation the protoassociativity of the generalized scan operations described here.

The algorithms given above to solve a tridiagonal system can be applied to banded positive-definite systems, with identical upper and lower bandwidth b, by covering the non-zero entries in the coefficient matrix with block tridiagonal matrices. Positive definiteness implies that the diagonal blocks in this covering are non-singular and that this property is inherited by the reduced matrices.

Theorem 6. *An algorithm exists to solve any $n \times n$ banded linear system of upper and lower bandwidth b with $\mathrm{O}(nW(b)/b)$ work and $\mathrm{O}(T(b)\log n)$ parallel time where $W(b)$ and $T(b)$ are, respectively, the work and parallel time to invert a $b \times b$ non-singular matrix. Furthermore, this algorithm only requires that diagonal block matrices be invertible.*

However, the solution of banded systems can also be obtained without explicit use of the block repackaging of the matrix, but rather, by a direct application of the notion of generalized scan. This can be seen as follows.

For simplicity, we refer to matrices with identical lower and upper bandwidth b, although with no significant loss of generality. The suppression of an individual block, in the natural generalization of the described method to block matrices, can be emulated by the successive suppressions of the b indices pertaining to that block, in any order. (The inversion of a $b \times b$ matrix is replaced by b steps of Gaussian elimination.) The only important detail is that the index suppression is carried out by a generalized scan operator of adequate support, specifically with $\alpha = \beta = 2b - 1$. The inputs to the generalized-scan operator are now $(4b-1)$-tuples of scalar entries, rather than 4-tuples. Note, however (as can be easily seen), that only $3b$ of these entries are non-zero, and that each $4b$-tuple contains a total of $(b-1)$ zero entries (at its left and/or right end). Clearly, the work performed by this implementation matches the one performed by the block approach; its main interest lies in the fact that it avoids explicit matrix inversion and that it embodies an additional instance of generalized-scan computations.

5. Conclusions

We have introduced generalized scans, a new class of recursively defined length-preserving, sequence transformations that generalize the well-known prefix computations (scans). Generalized scans enjoy a property, called protoassociativity, that gives rise to ordinary associativity when they are specialized to ordinary scans. We show that the solution of positive-definite block tridiagonal linear systems can be cast as a generalized scan, thereby shedding light on the underlying structure that enables known parallelization schemes for this problem. We also describe a variety of work- and time-optimal parallel algorithms including some that are well known for tridiagonal systems and some that are much better suited to distributed computation.

References

[1] G.E. Blelloch, Scans as primitive parallel operations, IEEE Trans. Comput. 38 (1989) 1526–1538.
[2] J.J. Dongarra, A.H. Sameh, On some parallel banded system solvers, Parallel Comput. 1 (1984) 223–235.
[3] R.W. Hockney, A fast direct solution of poisson's equation using fourier analysis, JACM 12 (1965) 95–113.
[4] S.L. Johnsson, Solving narrow banded systems on ensemble architectures, Research Report YALEU/DCS/RR-418, Dept. of Computer Science, Yale University, New Haven, CT, 1985.
[5] S.L. Johnsson, Solving tridiagonal systems on ensemble architectures, SIAM J. Sci. Statist. Comput. 8 (1987) 354–392.
[6] R.E. Ladner, M.J. Fischer, Parallel prefix computation, JACM 27 (1980) 831–838.
[7] U. Meier, A parallel partition method for solving banded systems of linear equations, Parallel Comput. 2 (1985) 33–43.
[8] V. Pan, J.H. Reif, Efficient parallel solution of linear systems, Proc. 7th Ann. ACM Symp. on Theory of Computing, 1985, pp. 143–152.
[9] J.H. Reif, $O(\log^2 n)$ Time efficient parallel factorization of dense, sparse separable, and banded matrices, Proc. 6th Ann. ACM Symp. on Parallel Algorithms and Architectures, 1994 pp. 278–289.
[10] H.S. Stone, An Efficient Parallel algorithm for the solution of a tridiagonal linear system of equations, JACM 20 (1973) 27–38.
[11] P.N. Swarztrauber, A direct method for the discrete solution of separable elliptic equations, SIAM J. Numer. Anal. 11 (1974) 1136–1150.
[12] R.A. Sweet, A generalized cyclic-reduction algorithm, SIAM J. Numer. Anal. 11 (1974) 506–520.
[13] R.A. Sweet, A cyclic-reduction algorithm for solving block tridiagonal systems of arbitrary dimension, SIAM J. Numer. Anal. 14 (1977) 706–720.
[14] H.H. Wang, A parallel method for tridiagonal equations, ACM Trans. Math. Software 7 (1981) 170–183.